Appendix

Proof of Theorem 2

First recall the following theorem from Verma and Pearl (1992); Meek (1995).

Theorem 3 *Given a PDAG where all the v-structures are oriented, then the CPDAG can be obtained by repeatedly applying the following orientation rules:*

- R_1 *Orient* $b - c$ *into* $b \to c$ *whenever there is an arrow* $a \to b$ *such that* a *and* c *are nonadjacent.*
- R_2 *Orient* $a - b$ *into* $a \to b$ *whenever there is chain* $a \to c \to b$.
- R_3 *Orient* $a - b$ *into* $a \to b$ *whenever there are two chains* $a - c \to b$ *and* $a - d \to b$ *such that* c *and* d *are nonadjacent.*
- R_4 *Orient* $a - b$ *into* $a \to b$ *whenever there are two chains* $a - c \to d$ *and* $c \to d \to b$ *such that* c *and* b *are nonadjacent and* a *and* d *are nonadjacent.*

Proof of Theorem 2 (If:) First, note that there are no compelled edges directed away from X (which would make X a non-leaf). Next, note that we can compel each edge $S - X$ towards X one-by-one, each time checking if Theorem 3 compels another edge in $S' - X$ towards S' instead. If this never happens, then all edges $S - X$ can be oriented towards X (which makes X a possible leaf).

We start by orienting one edge $S - X$ towards S. Consider each of the rules in Theorem 3:

- R_1 cannot be used to orient the edge from $X \to S$. First, if there were some other compelled edge $Y \to X$ where $Y \notin \mathbf{S}$ and Y is not adjacent to S, then the edge $S - X$ should already have been compelled (i.e., P was not a CPDAG). Otherwise, we only orient edges from $S \to X$ and all $S \in \mathbf{S}$ are adjacent.
- R_2 cannot be used to orient the edge from $X \to S$, since there is no chain from X to S (which would imply X was a non-leaf).
- R_3 cannot be used to orient the edge from $X \to S$, since all potential neighbors c and d via an unoriented edge must be adjancent.
- R_4 cannot be used to orient the edge from $X \to S$, since all potential neighbors c and b via an unoriented edge must be adjancent.

Since no rule compels us to orient an edge away from X, we can orient the edges one-by-one to make X a leaf.

(Only if:) We show that if $\mathbf{S}$ is not a clique, then X is not a leaf. If $\mathbf{S}$ is not a clique, then there is a pair S and S' in $\mathbf{S}$ that are non-adjacent. If we orient $S - X$ as $S \leftarrow X$, then X is not a leaf. If we orient $S - X$ as $S \to X$, then by Theorem 3, Rule R_1, we must orient $S' - X$ as $S' \leftarrow X$. Hence, X is not a leaf. □

Proofs of Propositions

Proof of Proposition 1 Follows from the definition of equivalence classes. □

Proof of Proposition 2 This path can be constructed by following any path to G in the BN graph, and then identifying the corresponding CPDAGs in the EC graph. □

Proof of Proposition 3 Follows from Theorem 10 in Chickering (1995). □

Proof of Proposition 4 Follows from the fact that the canonical ordering is the ordering with the largest reverse lexicographic order. □

EXPERIMENTS: EC TREE VS. BN GRAPH

In this section, we provide additional experimental results on EC trees and BN graphs, to gain a deeper insight into their performances differences. We first enumerate the 10-best, 100-best, and 1000-best equivalence classes using a EC tree. Using the BN graph, we then enumerate an equivalent number of DAGs, per dataset; Table 4 (from the main text) reports these numbers. Table 6 summarizes the time (in seconds) and memory (in GB) used by the EC tree and the BN graph, during A* search. As seen in the paper, the EC tree is more efficient (both in speed and memory consumption) than the BN graph, up to orders of magnitude differences (at least in time).

Table 7 shows, for the BN graph, the number of generated nodes, expanded nodes, re-expanded nodes (by partial-expansion), and the number of invocations of the oracle. In contrast to the EC tree (from Table 2 in the main text), the BN graph usually expands and generates at least one order of magnitude more nodes, and in some datasets, e.g., letter, msnbc and nltcs, more than three orders of magnitudes. In addition, Table 8 shows the time spent on computing the heuristic function T_h and the time spent on traversing the search space T_{A*}. For the data sets where the k-best equivalence classes contains a large number of DAGs, i.e., adult, letter, msnbc and nltcs, the majority of the time

benchmark			10-best EC				100-best EC				1,000-best EC			
			EC tree		BN graph		EC tree		BN graph		EC tree		BN graph	
name	n	N	t	m	t	m	t	m	t	m	t	m	t	m
adult	14	30162	0.26	1	0.47	1	0.54	1	1.49	1	1.19	1	11.87	1
wine	14	178	0.05	1	0.09	1	0.15	1	0.33	1	0.86	1	3.89	1
nltcs	16	16181	3.36	1	11.34	1	5.56	1	46.91	1	9.58	1	1126.05	4
letter	17	20000	18.11	1	20.68	1	48.31	1	84.07	1	81.72	1	4666.29	4
msnbc	17	291326	145.71	1	896.45	2	153.05	1	$\times_t$		155.07	1	$\times_t$	
voting	17	435	1.89	1	2.86	1	2.11	1	3.70	1	6.67	1	17.89	1
zoo	17	101	2.89	1	5.09	1	3.59	1	5.85	1	6.03	1	10.34	1
statlog	19	752	29.28	1	51.89	1	41.99	1	73.77	1	43.89	1	76.99	1
hepatitis	20	126	36.33	1	86.05	2	63.37	1	176.83	2	101.05	2	284.46	4
imports	22	205	174.84	4	455.65	8	223.78	4	603.68	8	224.11	4	604.14	8
parkinsons	23	195	897.81	8	779.90	16	897.97	8	1034.50	16	898.68	8	1450.46	16

Table 6: Time t (in seconds) and memory m (in GBs) used by EC tree and BN graph. n is the number of variables in the dataset, and N is the size of the dataset. A $\times_t$ corresponds to an out-of-time (2hr).

benchmark		10-best EC				100-best EC				1,000-best EC			
name	n	gen.	exp.	re-exp.	invoke	gen.	exp.	re-exp.	invoke	gen.	exp.	re-exp.	invoke
adult	14	1672	1352	3852	173	30619	27015	179312	634	245687	215753	2602353	1050
wine	14	8203	3714	0	107	44209	23572	23124	274	461799	254154	500024	595
nltcs	16	56719	53572	452232	633	326813	314528	6021330	1372	2727808	2605978	82512570	2180
letter	17	16296	16227	153430	678	230931	230931	4948105	2726	5443620	5424968	213643716	4963
msnbc	17	1288695	1288339	10564990	2695	$\times_t$				$\times_t$			
voting	17	5314	4364	346	147	72658	50004	99182	413	114498	106378	421512	3965
zoo	17	1049	704	0	330	12003	3875	3498	539	53562	47162	41698	1695
statlog	19	1915	1558	0	212	19653	12847	12403	628	153048	101570	194334	1029
hepatitis	20	18726	15470	0	4645	167033	105997	105105	13223	1378039	720381	1422924	31854
imports	22	1023	295	0	150	8041	2724	0	404	41007	21475	0	426
parkinsons	23	8233	2732	0	290	32136	10189	0	652	151200	58802	0	1273

Table 7: BN graph: number of nodes (1) generated, (2) expanded, (3) re-expanded, and (4) oracle invocations.

benchmark		10-best EC		100-best EC		1,000-best EC	
name	n	T_h	T_{A*}	T_h	T_{A*}	T_h	T_{A*}
adult	14	0.46	0.02	0.88	0.61	1.11	10.76
wine	14	0.05	0.04	0.06	0.26	0.12	3.77
nltcs	16	9.22	2.11	15.00	31.90	18.23	1107.81
letter	17	19.52	1.16	48.28	35.79	70.22	4596.07
msnbc	17	126.34	770.11	$\times_t$		$\times_t$	
voting	17	2.80	0.06	2.86	0.84	7.32	10.57
zoo	17	5.06	0.02	5.77	0.09	9.69	0.65
statlog	19	51.78	0.11	73.47	0.30	75.31	1.68
hepatitis	20	85.56	0.48	174.07	2.76	263.28	21.18
imports	22	454.71	0.93	602.49	1.19	602.71	1.43
parkinsons	23	778.45	1.45	1032.56	1.94	1447.56	2.89

Table 8: Time T_h to compute the heuristic function and time T_{A*} spent in A* search, in the BN graph (in seconds).

is spent on exploring the search space, rather the computing the heuristic function. This is in contrast to the EC tree, illustrated in Table 3, and the smaller datasets in Table 8, where the heuristic function is the performance bottleneck. However, on the EC tree, for these larger datasets, only a very small amount of time is used to traverse the search space (in Table 3), which shows that the EC tree is a more compact and efficient search space for enumerating equivalence classes.